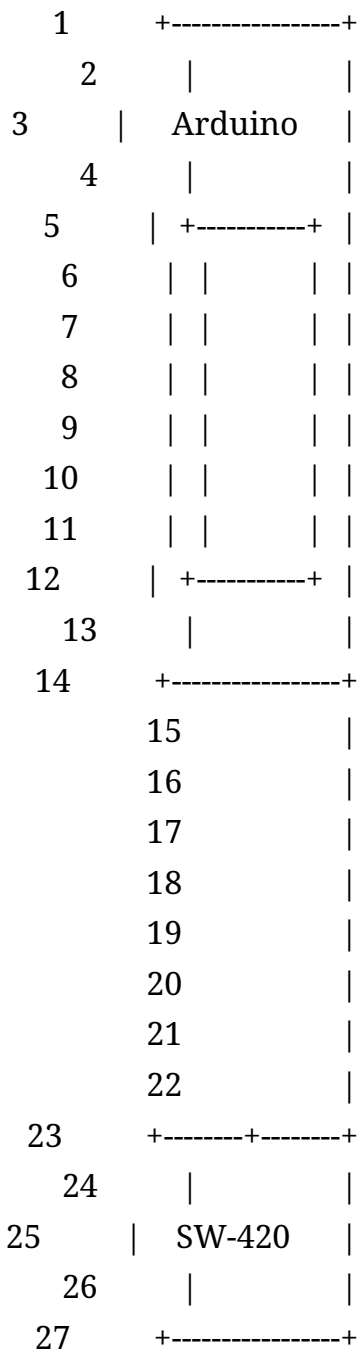
1º Objetivo: Esquemático do Projeto e Protótipo Inicial (1 Sensor)

Esquemático do Hardware

Vamos começar com o sensor de vibração SW-420. O circuito básico é simples e pode ser representado da seguinte forma:

Diagrama do Circuito:

plaintext
VerifyOpen In EditorEditCopy code



Conexões:

- VCC do SW-420 -> 5V do Arduino
- GND do SW-420 -> GND do Arduino
- DO do SW-420 -> Pino digital 2 do Arduino

Protótipo Inicial Funcional

Código para Captura de Dados do Sensor SW-420:

cpp
VerifyOpen In EditorEditCopy code

```
1const int vibrationPin = 2; // pino do sensor de vibração
2int vibrationState = 0;
3
4void setup() {
5  Serial.begin(115200);
6  pinMode(vibrationPin, INPUT);
7}
8
9void loop() {
10  // Lê o estado do sensor de vibração
11  vibrationState = digitalRead(vibrationPin);
12
13  // Envia o estado do sensor para o monitor serial
14  Serial.print("Vibration State: ");
15  Serial.println(vibrationState);
16
17  delay(500); // Ajuste conforme necessário
18}
```

Teste e Documentação:

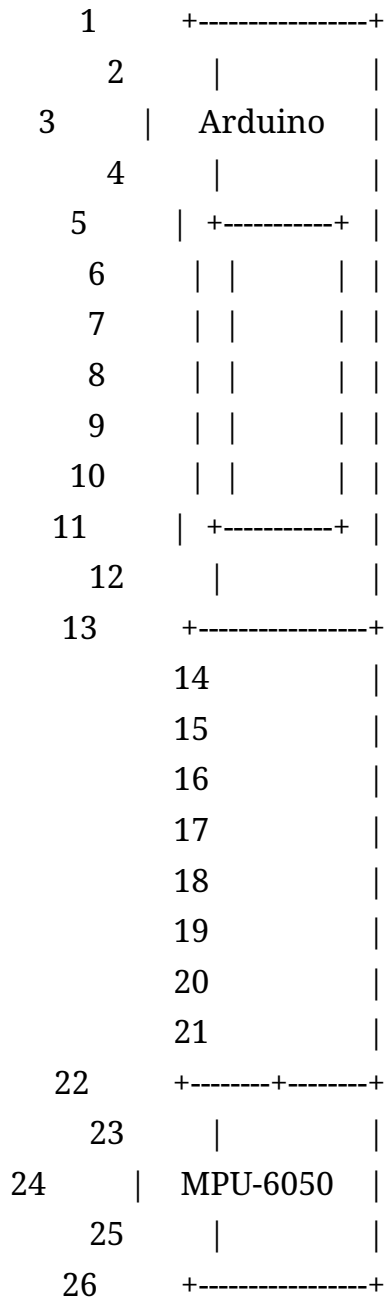
1. Monte o circuito conforme o diagrama.
2. Carregue o código acima no Arduino.
3. Abra o monitor serial no Arduino IDE para visualizar a saída, que mostrará se o sensor está detectando vibrações (1) ou não (0).

2º Objetivo: Sensor Principal Implementado e Modelo Físico Criado

Implementar o Sensor Principal (MPU-6050)

Diagrama do Circuito com MPU-6050:

plaintext
VerifyOpen In EditorEditCopy code



Conexões do MPU-6050:

- VCC -> 5V do Arduino
- GND -> GND do Arduino
- SDA -> A4 do Arduino

- SCL -> A5 do Arduino

Código para Captura de Dados do MPU-6050

cpp
VerifyOpen In EditorEditCopy code

```
1#include <Wire.h>
2#include <MPU6050.h>
3
4MPU6050 mpu;
5
6void setup() {
7  Serial.begin(115200);
8  Wire.begin();
9  mpu.initialize();
10}
11
12void loop() {
13  int16_t ax, ay, az;
14  mpu.getAcceleration(&ax, &ay, &az);
15
16  Serial.print("AX: ");
17  Serial.print(ax);
18  Serial.print(", AY: ");
19  Serial.print(ay);
20  Serial.print(", AZ: ");

 Serial.println(az);
22
23  // Adicione lógica para detectar mudanças bruscas de aceleração
24  if (abs(ax) > 1000 || abs(ay) > 1000 || abs(az) > 1000) {
25    Serial.println("Impact detected!");
26  }
27
28  delay(500);
29}
```

Modelo Físico do Sistema

Monte o protótipo físico com o MPU-6050 conectado ao Arduino. Utilize um protoboard e fios para facilitar as conexões. Documente o circuito final com um diagrama detalhado, utilizando softwares como Fritzing ou desenhando manualmente.

3º Objetivo: Todo o Hardware Implementado

Integração de Todo o Hardware

1. **Conexões Finais:**
    a. Conecte o MPU-6050 e o SW-420 ao Arduino Uno conforme descrito anteriormente.
    b. Certifique-se de que o Arduino está corretamente alimentado e que todas as conexões estão firmes.
2. **Configuração da Comunicação:**
    a. Conecte o Arduino Uno ao ESP32 conforme mencionado anteriormente, garantindo que a comunicação serial esteja configurada corretamente.

Funcionamento Completo

Código para Identificação de Impactos e Vibrações:

cpp
VerifyOpen In EditorEditCopy code

```
1#include <Wire.h>
2#include <MPU6050.h>
3
4MPU6050 mpu;
5
6const int vibrationPin = 2; // pino do sensor de vibração
7int vibrationState = 0;
8
9void setup() {
10  Serial.begin(115200);
11  Wire.begin();
12  mpu.initialize();
13  pinMode(vibrationPin, INPUT);
14}
15
```

```
16void loop() {
17  // Lê os dados do MPU-6050
18  int16_t ax, ay, az;
19  mpu.getAcceleration(&ax, &ay, &az);
20
21  // Lê o estado do sensor de vibração
22  vibrationState = digitalRead(vibrationPin);
23
24  // Envia os dados para o ESP32
25  Serial.print("AX: ");
26  Serial.print(ax);
27  Serial.print(", AY: ");
28  Serial.print(ay);
29  Serial.print(", AZ: ");
30  Serial.print(az);
31  Serial.print(", Vibration: ");
32  Serial.println(vibrationState);
33
34  // Detecta impactos
35  if (abs(ax) > 1000 || abs(ay) > 1000 || abs(az) > 1000) {
36    Serial.println("Impact detected!");
37  }
delay(500); // Ajuste conforme necessário
40}
```

Documentação Final

1. **Diagrama Final:**
    a. Crie um diagrama que mostre todas as conexões do hardware, incluindo o Arduino, os sensores e o ESP32. Utilize softwares como Fritzing para facilitar a visualização.
2. **Testes Realizados:**
    a. Documente todos os testes realizados, incluindo a resposta dos sensores a diferentes tipos de vibrações e impactos. Registre os resultados no monitor serial e faça anotações sobre o comportamento do sistema.
3. **Resultados:**

a. Compile os dados coletados e analise se o sistema responde como esperado. Se necessário, ajuste os parâmetros de sensibilidade e a lógica de detecção para melhorar a precisão do sistema.

Com essas etapas, você terá um sistema IoT funcional que captura dados de sensores e os envia para uma aplicação web, permitindo monitorar vibrações e impactos em tempo real. ### 1º Objetivo: Esquemático do Projeto e Protótipo Inicial (1 Sensor)